1 Manual de Usuario

Manual de usuario de la Interfaz de Aplicaciones de Programación ICS-API, que proporciona los mecanismos para poder comunicarse con la Arquitectura de Sistemas I8K. Ambos productos desarrollados en el presente PFC.

ICS-API puede ser utilizada por dos tipos de aplicaciones:

- Aplicaciones que desean realizar consultas a proveedores de datos (Tipo de organización *AP*).
- Proveedores de datos (Tipo de organización *Provider*).

Para descargarse ICS-API, las aplicaciones tienen que acceder a la dirección Web http://alarcosj.esi.uclm.es/i8k, donde visualizarán la Figura 1.

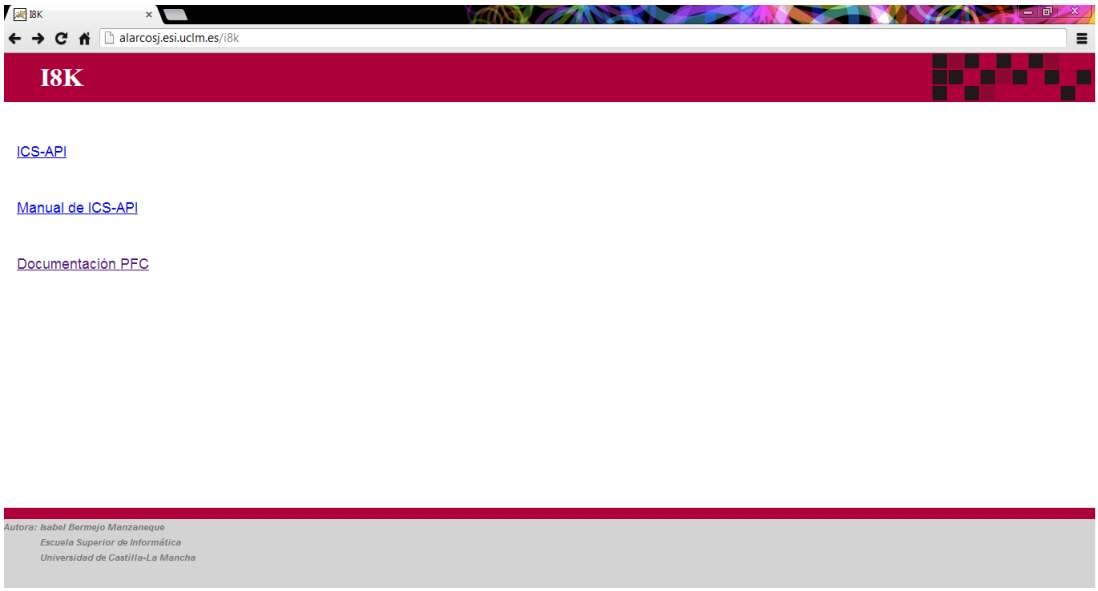

Figura 1: Interfaz de la página web.

- En el enlace **ICS-API** se descarga el paquete ejecutable ICS-API-1.3.jar. Este ejecutable es el que las aplicaciones tienen que incorporar a sus proyectos para hacer uso de la Arquitectura de Servicios I8K.
- En el enlace **Manual ICS-API** se descarga el presente manual de usuario.
- En el enlace **Documentación PFC** se descarga toda la documentación que compone este PFC.

A continuación se detallarán cada uno de los pasos que debe seguir cada una de las aplicaciones que pueden utilizar ICS-API.

1.1 Aplicación del tipo AP

1.1.1 Codificar un Mensaje de Datos Maestros

1. Descargar e importar el fichero ICS-API-1.3.jar en su proyecto.

2. Crear un objeto del tipo MDQManager. Con este objeto se realizan todas las funcionalidades que ofrece ICS-API.

MDQManager manAP=new MDQManager();

3. Establecer los parámetros de configuración para ICS-API. Para ello debe invocar el método *configureOrganization* y como parámetros de entrada suministrarle su nombre de organización y el tipo (AP).

manAP.configureOrganization("TripPlanner", OrganizationType.AP);

4. Establecer quién es el proveedor al que le va a enviar el mensaje. Para ello invoque el método *setDestination* y como parámetro de entrada le suministra el nombre del proveedor.

manAP.setDestination("HotelSA");

5. Para **codificar** un mensaje de petición de datos a un prooveedor debe seguir los siguientes pasos:

   (a) Identificar los términos que desea enviar en el mensaje. Para ello debe invocar al método *addTermAndValue* y como parámetros introducir el nombre del término y el valor o valores asociados[1].

   manAP.addTermAndValue("FROM_LOCATION", "Madrid");

   (b) Finalmente debe invocar el método *encodeAndCertificated*. Esté método le devuelve el mensaje codificado y listo para enviárselo al proveedor.

   message=manAP.encodeAndCertificated();

[1] Hay términos que pueden tener rangos de valores, por ejemplo el término DEPARTURE_DATE puede tener una lista de fechas en las que se desea salir del origen

1.1.2 Codificar y certificar un Mensaje de Datos Maestros

1. Descargar e importar el fichero ICS-API-1.3.jar en su proyecto.

2. Crear un objeto del tipo MDQManager. Con este objeto se realizan todas las funcionalidades que ofrece ICS-API.

MDQManager manAP=new MDQManager();

3. Establecer los parámetros de configuración para ICS-API. Para ello debe invocar el método *configureOrganization* y como parámetros de entrada sunimistrale su nombre de organización y el tipo (AP).

manAP.configureOrganization("TripPlanner", OrganizationType.AP);

4. Establecer quién es el proveedor al que le va a enviar el mensaje. Para ello invoque el método *setDestination* y como parámetro de entrada le suministra el nombre del proveedor.

manAP.setDestination("HotelSA");

5. Establecer los parámetros de certificación si desea recibir los datos contenidos en el mensaje que le responda el proveedor certificados. Para ello debe invocar el método *configureCertification* y como parámetros de entrada suminístrele el nivel mínimo requerido de precisión en los datos, nivel mínimo requerido de compleción en los datos, y si desea certificar la precisión y/o la compleción.

manAP.configureCertification(0, 50, false, true);

6. Para **codificar** un mensaje de petición de datos a un prooveedor debe seguir los siguientes pasos:

   (a) Identificar los términos que desea enviar en el mensaje. Para ello debe invocar al método *addTermAndValue* y como parámetros introducir el nombre del término y el valor o valores asociados[2].

[2]Hay términos que pueden tener rangos de valores, por ejemplo el término DEPARTURE_DATE puede tener una lista de fechas en las que se desea salir del origen

manAP.addTermAndValue("FROM_LOCATION", "Madrid");

(b) Si desea recibir el mensaje del proveedor con los datos certificados debe introducir las reglas de calidad de datos.

Para certificar el nivel de precisión:

i. Identificar los términos del proveedor que desea que el valor de ellos esté en un formato específico y si es obligatorio que cumpla dicho formato.

manAP.addTermPattern(ÇHECK_IN",
"Ĵ((19|20)dd)-(0?[1-9]|1[012])-(0?[1-9]|[12][0-9]|3[01])$", true);

ii. De igual forma identificar los términos del proveedor que desea comprobar que el valor existe y es correcto de acuerdo a alguna fuente.

manAP.addTermSource("DESTINATION",
"http://wordnet.princeton.edu/", true);

Para certificar el nivel de compleción:

(c) Identificar los términos del proveedor que desea que el valor este disponible.

manAP.addTermRequired("HOTEL_NAME", true);

(d) Finalmente debe invocar el método *encodeAndCertificated*. Esté método le devuelve el mensaje codificado y listo para enviárselo al proveedor.

message=manAP.encodeAndCertificated();

1.1.3 Decodificar un Mensaje de Datos Maestros

1. Descargar e importar el fichero ICS-API-1.3.jar en su proyecto.

2. Crear un objeto del tipo MDQManager. Con este objeto se realizan todas las funcionalidades que ofrece ICS-API.

MDQManager manAP=new MDQManager();

3. Establecer los parámetros de configuración para ICS-API. Para ello debe invocar el método *configureOrganization* y como parámetros de entrada sunimistrale su nombre de organización y el tipo (AP).

manAP.configureOrganization("TripPlanner", OrganizationType.AP);

4. Para **decodificar** un mensaje que ha recibido de un proveedorer debe seguir los siguientes pasos:

   (a) Invocar el método *decode* que recibe como parámetro el mensaje del proveedor.

   manAP.decode(message);

5. Para conocer los valores de los términos del mensaje del proveedor, debe invocar al método *getValue* que recibe como parámetro el término del que se desea conocer el valor.

1.2 Aplicación del tipo Provider

1.2.1 Codificar un Mensaje de Datos Maestros

1. Descargar e importar el fichero ICS-API-1.3.jar en su proyecto.

2. Crear un objeto del tipo MDQManager. Con este objeto se realizan todas las funcionalidades que ofrece ICS-API.

   MDQManager manPRO=new MDQManager();

3. Establecer los parámetros de configuración para ICS-API. Para ello debe invocar el método *configureOrganization* y como parámetros de entrada suminístrele su nombre de organización y el tipo (Provider).

   manPRO.configureOrganization("HotelSA", OrganizationType.Provider);

4. Establecer quién es la aplicación a la que enviará su respuesta. Para ello invoca al método *setDestination* y como parámetro de entrada le suministra el nombre de la aplicación.

   manPRO.setDestination("TripPlanner");

5. Para **codificar** un mensaje de respuesta de datos a una aplicación, debe seguir los siguientes pasos:

(a) Identificar los términos que desea enviar en el mensaje. Para ello debe invocar al método *addTermAndValue* y como parámetros introducir el nombre del término y el valor o valores asociados[3].

manPRO.addTermAndValue("DESTINATION", "Madrid");

(b) Invocar el método *encodeAndCertificated*. Esté método le devuelve el mensaje codificado y listo para enviarselo a la aplicación.

message=manPRO.encodeAndCertificated();

1.2.2 Decodificar un Mensaje de Datos Maestros

1. Descargar e importar el fichero ICS-API-1.3.jar en su proyecto.

2. Crear un objeto del tipo MDQManager. Con este objeto se realizan todas las funcionalidades que ofrece ICS-API.

MDQManager manPRO=new MDQManager();

3. Establecer los parámetros de configuración para ICS-API. Para ello debe invocar el método *configureOrganization* y como parámetros de entrada suminístrele su nombre de organización y el tipo (Provider).

manPRO.configureOrganization("HotelSA", OrganizationType.Provider);

4. Establecer quién es la aplicación a la que enviará su respuesta. Para ello invoca al método *setDestination* y como parámetro de entrada le suministra el nombre de la aplicación.

manPRO.setDestination("TripPlanner");

5. Para **decodificar** un mensaje que ha recibido de una aplicación debe seguir los siguientes pasos:

(a) Invocar el método *decode* que recibe como parámetro el mensaje de la aplicación.

[3]Hay términos que pueden tener rangos de valores, por ejemplo el término CHECK_IN puede tener una lista de fechas en las que se desea entrar en el hotel

manPRO.decode(message);

6. Para conocer los valores de los términos del mensaje de la aplicación, debe invocar al método *getValue* que recibe como parámetro el término del que se desea conocer el valor.